Appendix A

Kernel

Major Components

This appendix details the major software components within the fraud detector application domain including analysis and design details required.

The following is a list of passive objects identified as part of the analysis phase which will now be described in more detail using the object numbers in parentheses:

- Fraud Detection Client (27)
- Interpret Call Detail Record (15)
- Add Knowledge Request (23)
- Update Historic Profile Request (24)
- Performance Evaluation Request (29)
- Fraud Detection Request (16)
- Poll To Recent Profile Decay (20)
- CDR To Profile Tranform (13)
- Call Detail Record (12)
- Unvalidated Fraud Candidates (25)
- Fraud Detector Specification (28)
- Validate Request (8)
- Candidate Data Set (18)
- Validated Fraud Candidate (22)
- Fraud Candidate (11)
- Presentation Data Set (17)
- Fraud Candidate Data Set (21)
- Profile Data Presentation (7)
- Poll Profile Vector (4)
- Recent Profile Vector (34)
- Historic Profile Vector (33)
-

Fraud Detection Client (27)

Description

A representation of a client of a fraud detector. This controls the fraud detection and performance evaluation requests of the application.

C++ class name
FDFraudDetectionClient

Behaviour Description: CreateFraudKernel
Upon receiving the CreateFraudKernel creation event from the GUI terminator, this object will:

- link to the specified fraud detector specification, object 28, which was passed as a parameter associated with the creation event.
- establish a clock polling mechanism.
- Read customer recent and historical profiles via the persistence mechanism (See Appendix B) creating a profile data presentation, object 7, for each individual customer and added to the presentation data set, object 17.
- The set of recent profiles is sent to construct poll to recent profile decay, object 20.
- A handle needs to be kept on both the presentation data set, object 17, and poll to recent profile decay, object 20.
- When the creation process is complete this object will send a KernelCreated event back to the GUI terminator.

The fraud detection client is now ready to service other events.

Behaviour Description: UpdateEvaluationInterval
Upon receiving an UpdateEvaluationInterval event from the GUI terminator the client will modify the no_evaluation_period attribute of the Fraud Detector Specification object (28) with the new evaluation interval.

Behaviour Description: UpdateDetectionStartDate
Upon receiving an UpdateDetectionStartDate event from the GUI terminator the client will modify the detection_start attribute of the Fraud Detector Specification object (28) with the new date. The client will then stop and update the poll clock mechanism with the new detection time and restart the poll clock mechanism.

Behaviour Description: UpdatePerformanceThreshold
Upon receiving an UpdatePerformanceThreshold event from the GUI terminator the client will modify the evaluation_performance attribute of the

Fraud Detector Specification object (28) with the new performance threshold.

Behaviour Description: AddKnowledge

Upon receiving an AddKnowledge event from the GUI terminator which contains a handle to a set of fraud candidate objects (11), the client will then create an AddKnowledgeRequest Object (23) with the associated fraud candidate set. On completion of the request the client will be informed by the AddKnowledgeRequest Object (23) what operations have been completed. These operations will be detailed by use of an enumeration parameter with an associated real value. The enumeration type contains the following:

- AddKnowledge
- PerformanceEvaluation
- Retraining

If the enumeration value is "AddKnowledge" then the associated real value will be zero, else it will indicate the current performance of the ADE. These values will then be used to send a AddKnowledgeComplete event to GUI terminator.

Behaviour Description: SwitchEngine

Upon receiving a SwitchEngine event from the GUI terminator the client will interrogate the event parameter to establish if a switch is required. If a switch is required then a request will be made to the ADE to switch to a new anomaly detector. If a switch is not required then no request is made of the ADE. On completion of the switch process the client will send a SwitchComplete event to the GUI terminator.

Note: The client is required to control the persistence of the new ADE on completion.

Behaviour Description: PollTime

Upon receiving a PollTime event from the Process IO (clock poll mechanism) terminator which indicates that a detection poll period has been reached. The client will send a DetectionTakingPlace to the GUI terminator to indicate that the client cannot except any events until the

operation has been completed. The client will create a fraud detection request object (16) which will control the detection process. On completion the client will send a DectionResultsReady event to GUI terminator. This event includes the time stamp used to create the results file.

Note: If the kernel is busy when a poll detection period is reached then when the client becomes available it will get the current time. If this time is less than the clock interval (plus some overhead time) then the detection is serviced else the poll detection has been missed and the kernel sends a DetectionMissed message back to the GUI to indicate that a poll detection has been missed.

Methods

FDFraudDetectionClient (FDFraudDetectorSpecification& fraud_spec)
~FDFraudDetectionClient()

static FDFraudDetectionClient* CreateFraudKernel (FDFraudDetectorSpecification& fraud_spec)

void UpdateEvaluationInterval(int evaluation_interval)

void UpdateDetectionStartDate(date detection_date)

void UpdatePerformanceThreshold(
float performance_threshold)

void AddKnowledge(FDFraudCandidateDataSet& data_set)

void SwitchEngine(Bool switch_required)

void PollTime()

Assumptions

- The bridge will create fraud detector specification object on CreateFraudKernel.
- The bridge will create fraud candidate date set object hierarchy on
- AddKnowledge.
- Retraining will always result in an improved performance of the ADE.

- Retraining can follow a retraining without a SwitchEngine event being received.

Ownership
FDFraudDetectorSpecification
FDAddKnowledgeRequest
FDFraudDetectionRequest

Read Accessors
RWBoolean IsAnomalyDetectorCreated() const;
FDPresentationDataSet* GetPresentationDataSet() const;
RWBoolean GetADSwitched() const;

Write Accessors
void SetADSwitched(RWBoolean state);

Interpret Call Detail Record (15)
Description
The transformation that is required in order to interpret a comma separated CDR into a CDR.

Note: Not implemented, absorbed into Validate Request (8).

Add Knowledge Request (23)
Description
A request to add knowledge of fraud candidates.

C++ class name
FDAddKnowledgeRequest

Behaviour Description
Upon creation the add knowledge request object (23) is passed a fraud detection data set as a parameter. The object will:

- Sends an APP6AddKnowledge event to the ADE terminator including the set of example detection data presentations, object (9), contained within the specified data set. These should only include those account which have been validated (For more information see "Enumeration Types" on page 53.).

- Upon completion the ADE generates an APP14KnowledgeAdded, which contains a handle to the new knowledge set. This object must persist this information using the new_knowledge_filename.
- create a update historic profile request, object 24, attaching the specified data set.
- check if a performance update is required by interrogating the performance evaluation counter attribute of the fraud detection client, object (27), and determining if it equals the number of evaluations specified contained within the fraud detector specification, object (28). If a performance update is required then a performance evaluation request is created and the performance evaluation counter attribute is reset to zero. If a performance update is not required then the performance evaluation counter attribute is incremented.

The operation enumeration is set to "AddKnowledge" as default.

Methods
FDAddKnowledgeRequest(
FDFraudCandidateDataSet& fraud_data_set,
String new_knowledge_filename)
~FDAddKnowledgeRequest()

Assumptions
Update Historic Profile Request (24) will always be actioned after an Add Knowledge Request (23).

Ownership
FDUpdateHistRequest
FDPerformanceEvaluationRequest

Read Accessors
No public read access methods are required by the object.

Write Accessors
No public write access methods are required by the object

Update Historic Profile Request (24)
Description

A request to update historic profiles.

C++ class name
FDUpdateHistRequest

Behaviour Description
Upon creation the update historic profile request is passed a fraud detection data set as a parameter. This object will:

- Sends an APP7UpdateHistoricProfiles event to the ADE terminator including the set of profile data presentations. Only those validated fraud candidates with a validation category of either; correct non-fraudulent or incorrect fraud candidates. In addition all the other non-fraud candidates are passed to the ADE.
- Upon completion the ADE generates an APP15ProfilesUpdated, the event contains the updated profiles. The update historic profiles request then needs to persist all the updated historical profiles. This data set can then be removed.

Methods
FDUpdateHistRequest(
FDFraudCandidateDataSet& fraud_data_set,
String historic_profile_filename)
~FDUpdateHistRequest()

Assumptions
None.

Ownership

Read Accessors
No public read access methods are required by the object.

Write Accessors
No public write access methods are required by the object

Performance Evaluation Request (29)
Description

A request to evaluate the performance of the fraud detector application.

C++ class name
FDPerformanceEvaluationRequest

Behaviour Description
No parameters are sent on construction of this object. This object will:

- Sends an APP3EvaluatePerformance event to the ADE. Upon completion the ADE generates an APP11PerformanceResultsObtained event with the ADE current performance.
- If the resulting performance evaluation is less than the evaluation threshold attribute of the fraud detector specification then the performance evaluation request sends an APP4TrainAD event to the ADE. Upon completion the ADE generates an APP12AnomalyDetectorTrained with the a new performance from the ADE.
- The operation enumeration type object attribute of the add knowledge request needs to be set to either "PerformanceEvaluation" or "Retraining" to indicate which operation has been performed.
- The new performance is returned to the add knowledge request object.

Methods
FDPerformanceEvaluationRequest()
~FDPerformanceEvaluationRequest()

Assumptions
None.

Ownership

Read Accessors
No public read access methods are required by the object.

Write Accessors
No public write access methods are required by the object

Description

A request to perform a detection of fraud on a presentation data set. The resultant fraud candidates are contained in the associated candidate data set.

C++ class name

FDFraudDetectionRequest

Behaviour Description

Upon creation the fraud detection request is passed a presentation data set as a parameter. This object will:

- Creates CDR to profile tranform, object 13, with csv filename and poll detection period.
- CDR to profile tranform, object 13, returns a list of poll detection profiles, object 4.
- Creates fraud candidate, object 11, to be populated with the results from the ADE.
- Sends an APP2PerformDetection event to the ADE terminator, with profile data presentations, object 7, where the profile modified attribute is true.
- Once the ADE has completed the detection event the ADE generates an APP10DetectionComplete. The fraud candidate, object 11 is populated with candidate presentations, object 6, matching with the associated recent profile, object 4.
- The profile modified attribute within profile data presentation, object 7, for all those sent to the ADE terminator need to be set back to false.
- The fraud candidate, object 11, persistence mechanism to write the results to a file. The time stamp at time of creation of this file needs to added to the top of the file and maintained to be sent back to the client, object 27.
- Once the results file has been created the fraud candidate, object 11, can be removed.

CDR Extraction, Poll Profile Creation and Search Algorithm

```
while(not end_of_file)
{
Read(next_line_of_file)
cdr = CreateCDR(next_line_of_file)
```

```
if(account_no != cdr.account_no)
poll_profile = CreatePollProfile(cdr)
else
poll_profile = AccumulatePollProfile(cdr)
account_no = cdr.account_no

DecayRecent(poll_profile)
DeletePollProfile(poll_profile)
}
```

Note: Assumption that the CDR file is sorted by account number. Decay profile will provide a binary search technique to locate the recent profile.

Methods

```
FDFraudDetectionRequest(
FDPresentationDataSet& presentation_data_set
FDPollToRecentProfileDecay& profile_decay
String results_filename,
String csv_filename
Time poll_detection_period
Time recent_profile_period)
~FDFraudDetectionRequest()
```

Assumptions

None.

Ownership

Read Accessors

No public read access methods are required by the object.

Write Accessors

No public write access methods are required by the object

Poll To Recent Profile Decay (20)

Description

The decay transform for decaying a poll period profile into a recent profile.

C++ class name

FDPollToRecentProfileDecay

Behaviour Description

Upon creation this object is given recent profile vectors object (4). This object will:

- Create relationships to all recent profiles.
- Calculate update factor using poll detection period for source and recent profile period for target.
- Upon a DecayProfile event search for the corresponding recent profile. If no recent profile exists create new recent profile.
- Update the target profiles behaviour with the source target behaviour using the algorithm below.
- Once the recent profile has been updated the poll detection profile can be removed.
- Modifies the profile modified attribute within the associated profile data presentation, object 7, to true.

Methods

FDPollToRecentProfileDecay(
RWTPtrDlist<FDRecentProfileVector>& recent_profile,
Time poll_detection_period,
Time recent_profile_period)
~FDPollToRecentProfileDecay()

void DecayProfile(FDProfileVector& poll_profile)

Assumptions

None.

Updating profiles algorithm

$$T_i' = (T_i - (T_i x UpdateFactor)) + (S_i x UpdateFactor)$$

For all i Where T is the target profile (e.g. recent profile) and S is the source profile (e.g. poll detection period profile.)

$$UpdateFactor = \frac{WindowSize(S)}{WindowSize(T)}$$

Read Accessors

No public read access methods are required by the passive object.

Write Accessors

No public write access methods are required by the passive object

CDR To Profile Tranform (13)

Description

A request to perform a detection of fraud on a presentation data set. The resultant fraud candidates are contained in the associated candidate data set.

C++ class name

FDCDRProfileTranform

Behaviour Description

Upon creation CDR profile transform. This object will:

- For each call detail record, object 12, this object either constructs a poll profile, object 4, or updates the existing poll profile.
- This object sends the poll detection profile to poll to recent profile decay, object 20, with poll detection period and recent profile period.

Methods

FDCDRProfileTranform(String csv_filename, int poll_detection_period)

~FDCDRProfileTranform()

Assumptions

Operates on an ordered input file.

Ownership

FDProfileVector (Poll detection profiles only).

Read Accessors

No public read access methods are required by the passive object.

Write Accessors

No public write access methods are required by the passive object

Call Detail Record (12)
Description
A software representation of a telecommunication call detail record.

C++ class name
FDCallDetailRecord

Methods
FDCallDetailRecord(String csv_filename)
~FDCallDetailRecord()

FDCallDetailRecord ReadCallDetailRecord()

Assumptions
The source CDR file is ordered by account number.

Ownership

Read Accessors

Write Accessors

Unvalidated Fraud Candidates (25)
Description
An unvalidated association of a customers recent profile and the results of a detection process.

C++ class name
FDUnvalidatedFraudCandidates

Inheritance
FDFraudCandidate

Methods
FDUnvalidatedFraudCandidates(
FDProfileVector& recent_profile,
ADCandidatePresentation& candidate_presentation)

~FDUnvalidatedFraudCandidates()

Assumptions
None.

Ownership
None.

Read Accessors
No public read access methods are required by the passive object.

Write Accessors
No public write access methods are required by the passive object

Fraud Detector Specification (28)
Description
The specification of the fraud detector application.

C++ class name
FDFraudDetectorSpecification

Methods
FDFraudDetectorSpecification(String Default_results_filename
String csv_filename
String recent_profile_filename
String historical_profile_filename
String ade_spec_filename
Date detection_start
int evaluation_interval
int evaluation_counter
int performance_threshold
int recent_window_size
int historical_window_size
int detection_time_interval
int input_size
int recent_size)
~FDFraudDetectorSpecification()

Assumptions
None.

Ownership
None.

Read Accessors
StringGetDefaultResultsFilename(
default_results_filename)
String GetCSVFilename(csv_filename)
String GetRecentProfileFilename(
recent_profile_filename)
String GetHistoricalProfileFilename(
historical_profile_filename)
String GetADESpecFilename (ade_spec_filename
Date GetDetectionStart(detection_start)
int GetEvaluationInterval(evaluation_interval)
int GetEvaluationCounter(evaluation_counter)
int GetPerformanceThreshold(performance_threshold)
int GetHistoricalWindowSize(historical_window_size)
int GetRecentWindowSize(recent_window_size)
int GetDetectionTimeInterval(detection_time_interval)
int GetInputSize(input_size)
int GetRecentSize(recent_size)

Write Accessors
void SetDefaultResultsFilename(String default_results_filename)
void SetCSVFilename(String csv_filename)
void SetRecentProfileFilename(String recent_profile_filename)
void SetHistoricalProfileFilename(String historical_profile_filename)
void SetADESpecFilename (String ade_spec_filename
void SetDetectionStart(Date detection_start)
void SetEvaluationInterval(int evaluation_interval)
void SetEvaluationCounter(int evaluation_counter)
void SetPerformanceThreshold(int performance_threshold)
void SetHistoricalWindowSize(int historical_window_size)
void SetRecentWindowSize(int recent_window_size)
void SetDetectionTimeInterval(int detection_time_interval)

void SetInputSize(int input_size)
void SetRecentSize(int recent_size)

Validate Request (8)
Description
A request to create a validated set of fraud candidates.
Note: Not implemented, absorbed into Fraud Detection Request (16).

Candidate Data Set (18)
Description
A set of candidate presentations.

C++ class name
FDCandidateDataSet

Methods
FDCandidateDataSet(
RWTPtrDlist<ADCandidatePresentation>
&candidate_presentation_ids)
~FDCandidateDataSet()

Assumptions

Ownership

Read Accessors
int GetNumberOfPresentations() const;

Write Accessors
void SetNumberOfPresentations(int number_of_presentations);

Validated Fraud Candidate (22)
Description
An association of a customers recent profile and the validated results of a detection process.

C++ class name
FDValidatedFraudCandidate

Inheritance
FDFraudCandidate

Methods
FDValidatedFraudCandidate(
FDProfileVector& recent_profile,
NNExampleDataPresentation& example_presentation);
~FDValidatedFraudCandidate()

Enumeration Types
enum ValidationStatus
{
UNVALIDATED,
CORRECT_FRAUD,
INCORRECT_FRAUD,
CORRECT_NONFRAUD,
INCORRECT_NON_FRAUD
};

Assumptions
None.

Ownership

Read Accessors
ValidationStatus GetValidationCategory() const;

Write Accessors
void SetValidationCategory(ValidationStatus
validation_category);

Fraud Candidate (11)
Description
An association of a customers recent profile and the results of a detection process, (either validated or unvalidated).

C++ class name

FDFraudCandidate

Methods
FDFraudCandidate(FDProfileVector& recent_profile)
~FDFraudCandidate()

Assumptions

Ownership

Read Accessors
No public read access methods are required by the passive object.

Write Accessors
No public write access methods are required by the passive object

Presentation Data Set (17)
Description
A set of profile data presentations.

C++ class name
FDPresentationDataSet

Methods
FDPresentationDataSet(FDProfileDataPresentation&
profile_data_presentation_id)
FDPresentationDataSet(
RWTPtrDlist<FDProfileDataPresentation>&
profile_data_presentation_ids)
~FDPresentationDataSet()

Assumptions

Ownership

Read Accessors
int GetNumberOfPresentations() const;

Write Accessors
void SetNumberOfPresentations(int number_of_presentations);

Fraud Candidate Data Set (21)
Description
A container of fraud candidates.

C++ class name
FDFraudCandidateDataSet

Methods
FDFraudCandidateDataSet()
~FDFraudCandidateDataSet()

Assumptions

Ownership

Read Accessors
int GetNumberOfPresentations() const;

Write Accessors
void SetNumberOfPresentations(int number_of_presentations);

Profile Data Presentation (7)
Description
Combination of a historic and a recent profile data vector.

C++ class name
FDProfileDataPresentation

Behaviour Description
Each recent profile is matched with it respective historical profiles and sent to the ADE. This representation is used for both detection (object 16) and profile decay (object 24).

Methods
FDProfileDataPresentation(

FDProfileVector& recent_profile,
FDProfileVector historical_profile)
FDProfileDataPresentation(
FDProfileVector& recent_profile,
RWTPtrDlist<FDProfileVector>& historical_profile)
~FDProfileDataPresentation()

Assumptions
None.

Ownership

Read Accessors
Bool GetProfileModified() const;

Write Accessors
void SetProfileModified(Bool profile_modified);

Poll Profile Vector (4)
Description
Describes the structure of a profile data vector.

C++ class name
FDPollProfileVector

Inheritance
NNVector

Methods
FDPollProfileVector(String account_number,
FDCallDetailRecord& call_detail_record)
~FDPollProfileVector()

Assumptions

Ownership

Read Accessors

String GetAccountNumber() const;

Write Accessors
void SetAccountNumber(String account_number);

Recent Profile Vector (34)
Description
Describes the structure of a recent profile data vector.

C++ class name
FDRecentProfileVector

Inheritance
ADRecentProfileVector

Behaviour Description
- After the poll profiles have been used to update the recent profile, the updated recent profiles then needs to be persisted to the recent profile file using the persistence mechanism.

Methods
FDRecentProfileVector(String account_number,
NNVector& data_vector)
~FDRecentProfileVector()

Persist(String recent_profile_filename)

Assumptions

Ownership

Read Accessors
String GetAccountNumber() const;

Write Accessors
void SetAccountNumber(String account_number);

Historic Profile Vector (33)

Description
Describes the structure of a profile data vector.

C++ class name
FDHistoricProfileVector

Inheritance
ADHistoricalProfileVector

Methods
FDHistoricProfileVector(String account_number,
NNVector& data_vector)
~FDHistoricProfileVector()

Assumptions

Ownership

Read Accessors
String GetAccountNumber() const;

Write Accessors
void SetAccountNumber(String account_number);